# Aplicaţ ii backtraking pentru funcţii injective, surjective, bijective

## Funcţii injective. Aranjamente.

*Definiţie*

Fie f : A→ B o funcţie. f este o funcţie injectivă sau f este o injecţie, dacă pentru oricare două elemente x şi y ale lui A, x≠y, f(x)≠f(y).

Să se genereze toate funcţiile injective f :{a1,a2…..a**m**}->{b1,b2…..b**n**}, unde card(A)=m, card(B)=n

**O soluţie este formată din m elemente (nr.elementelor mulţimii A) iar fiecare element din soluţie reprezintă valoarea funcţiei f(k) (un element din B).**

Problema se reduce la ***generarea tuturor Anm*** (n>=m)

**Ex.1[[1]](#footnote-1).** Să se genereze toate funcţiile injective f :{a1,a2…..am}->{b1,b2…..bn} cu proprietatea că f(ai)\*f(ai+1) >= 0 .

Exemplu pentru a=(1,2,3) şi b=(-1,-2,3,-4,5)

Soluţiile (6) vor fi

|  |  |  |
| --- | --- | --- |
| k=3 | 4 | (1,2,3,4,5) |
| k=2 | 2 | (1,2,3,4,5) |
| k=1 | 1 | (1,2,3,4,5) **poziţia** elem.din **B** |

V[k]

Ex. V=(1,2,3) nu este soluţie

b[2]\*b[3]<0 ( -2\*3 <0 )

f(1)=-1 f(2)=-2 f(3)=-4

f(1)=-1 f(2)=-4 f(3)=-2

f(1)=-2 f(2)=-1 f(3)=-4

f(1)=-2 f(2)=-4 f(3)=-1

f(1)=-4 f(2)=-1 f(3)=-2

f(1)=-4 f(2)=-2 f(3)=-1

*Rezolvare :*

Presupunem că vectorul soluţiei este V, atunci orice V[k] reprezintă valoarea funcţiei f(k)

Pe nivel al stivei , V[k] va lua valori din {b1,b2…..b**n**}.Cum nu există o relaţie de ordine între elementele mulţimii B, atunci în V[k] vom reţine poziţia elementului din B.

Conform restricţiilor din enunţ validarea presupune ca în afara faptului că elementele trebuie să fie distincte trebuie să verificăm şi că f(ai)\*f(ai+1) >= 0 ; … pe orice nivel al stivei elementul este **invalid dacă (b[v[i]]\*b[v[i-1]]<0) sau v[k]==v[i] (i=1,k-1)**

## Funcţii surjective . Produs cartezian .

*Definiţie*

O funcţie f : A→ B este o funcţie surjectivă , sau f este o surjecţie, dacă pentru orice element bB există cel puţin un element aA, astfel încât f(a)=b.

Să se genereze toate funcţiile surjective f :{a1,a2…..a**m**}->{b1,b2…..b**n**}, unde card(A)=m, card(B)=n. O soluţie este formată din **m** elemente (nr.elementelor mulţimii A), fiecare soluţie .

Problema se reduce la ***generarea produsului cartezian B1xB2x….xBm ( Bm)\* (Atenţie !*** *algoritm exponenţial datorită numărului mare de elemente ale produsului cartezian* ***)***

**Vor fi considerate soluţii numai cele care conţin toate elementele din mulţimea B.(după generarea unei posibile soluţii verificaţi înainte de afişare !!!)**

Ex.2 . Să se genereze toate funcţiile surjective f :{a1,a2…..am}->{b1,b2…..bn} astfel încât

<=x, x număr întreg dat. Dacă problema nu are soluţie, se va afişa mesajul de eroare “Fără soluţie”.

|  |  |  |
| --- | --- | --- |
| k=3 | 2 | (1,2) |
| k=2 | 1 | (1,2) |
| k=1 | 1 | (1,2) |

V[k]

Pentru A=(1,2,3) B=(1,2) x=4 vor fi afişate funcţiile

f(1)=1 f(2)=1 f(3)=2 ( 1+1+2=4)

f(1)=1 f(2)=2 f(3)=1

f(1)=2 f(2)=1 f(3)=1

*Rezolvare :*

Presupunem că vectorul soluţiei este V, **orice V[k] va lua valori din B.**

Teoretic, pentru generarea produsului cartezian orice valoare este validă .

Restricţiile din enunţ , reduce numărul de soluţii.

Se consideră soluţie dacă <=x; Dacă nu se va găsi cel puţin o soluţie , atunci se va afişa mesajul corespunzător. (Ind. se pot număra soluţiile atunci când se afişează )

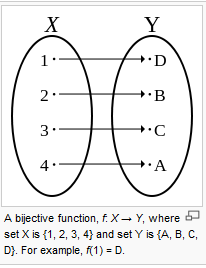
## Funcţii bijective . Permutări.

Definiţie :

O funcţie f : A→ B care este simultan injectivă şi surjectivă se numeşte funcţie bijectivă sau bijecţie.

Bijecţia (***sau corespondenţa de 1 :1***) este o funcţie în care card(A)=card(B) şi *fiecărui element din A îi corespunde exact un element din B*.

Dacă A=B , atunci putem spune că soluţia este o permutare a mulţimii A (sau B, evident).

[[2]](#footnote-2)

*Exemple practice pentru bijecţie* (traducere de la <http://en.wikipedia.org/wiki/Bijection> )

“ Într-o sală de clasă, există un anumit număr de locuri (scaune). Un grup de elevi intră în cameră și profesorul le cere tuturor să se așeze. După o scurtă privire în jurul camerei, profesorul declară că *este o functie bijectivă între setul de elevi și setul de scaune*, unde fiecare elev este asociat cu un scaun şi numai unul.

***La această concluzie a ajuns deoarece a observat că :***    Fiecare elev avea un scaun (nu era nimeni în picioare)

    Nici un elev nu avea mai mult de un scaun  
    Pe fiecare scaun stătea cineva (nu au existat locuri goale)  
    Pe un scaun stătea o singură persoană„

Aplicaţii.

1. Un grup de n (n<=10) persoane numerotate de la 1 la n sunt aşezate pe un rând de scaune, dar între oricare două persoane vecine s-au ivit conflicte. Scrieţi un program care afişează toate modurile posibile de reaşezare a persoanelor, astfel încât între oricare două persoane aflate în conflict să stea una sau cel mult două persoane.

Ex. Pentru n=4 programul ar trebui să afişeze:

3 1 4 2

2 4 1 3

1. Culegere probleme informatică Carmen Negrea/ Mihaela Runceanu Ed.Donaris 2003 [↑](#footnote-ref-1)
2. <http://en.wikipedia.org/wiki/Bijection> [↑](#footnote-ref-2)